OpenCMISS-iron examples and tests used by *OpenCMISS* developers at University of Stuttgart, Germany

Christian Bleiler*, Andreas Hessenthaler*, Thomas Klotz*, Aaron Krämer†, Benjamin Maier‡, Sergio Morales*, Mylena Mordhorst*, Harry Saini*

July 4, 2017
17:24

CONTENTS

* Institute of Applied Mechanics (CE), University of Stuttgart, Pfaffenwaldring 7, 70569 Stuttgart, Germany
† Institute for Parallel and Distributed Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
‡ Lehrstuhl Mathematische Methoden für komplexe Simulation der Naturwissenschaft und Technik, University of Stuttgart, Allmandring 5b, 70569 Stuttgart, Germany

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

This document contains information about examples used for testing *OpenCMISS-iron*. Read: How-to[1] and [1].

1.1 Cmgui files for cmgui-2.9

1.2 Variations to consider

- Geometry and topology
  - 1D, 2D, 3D
  - Length, width, height
  - Number of elements
  - Interpolation order
  - Generated or user meshes
  - quad/hex or tri/tet meshes
- Initial conditions
- Load cases
  - Dirichlet BC
  - Neumann BC
  - Volume force
  - Mix of previous items
- Sources, sinks
- Time dependence
  - Static
  - Quasi-static
  - Dynamic
- Material laws
  - Linear
  - Nonlinear (Mooney-Rivlin, Neo-Hookean, Ogden, etc.)
  - Active (Stress, strain)
- Material parameters, anisotropy
- Solver
  - Direct
  - Iterative
- Test cases
  - Numerical reference data
  - Analytical solution
- A mix of previous items

1.3 Folder structure

TBD..

1 https://bitbucket.org/hessenthaler/opencmiss-howto

2 HOW TO WORK ON THIS DOCUMENT

In the Google Doc at `https://docs.google.com/spreadsheets/d/1RGKj8vVPqQ-PH0UwMX_e9TAzqaYavKiOzOD4pKY9RGI/edit#gid=0` please indicate what you are working on or if a given example was finished

- no mark: to be done
- x: currently working on it
- xx: done

Initials	Full name
CB	Christian Bleiler
AH	Andreas Hessenthaler
TK	Thomas Klotz
AK	Aaron Krämer
BM	Benjamin Maier
SM	Sergio Morales
MM	Mylena Mordhorst
HS	Harry Saini

Table 1: Initials of people working on examples, in alphabetical order (surnames).

3 UNTIL FRIDAY

- Finish open examples
- Set a `TAG` for each example in document title:

  `DOCUMENTED`: finish the documentation of the example (spatial domain, number of time steps, boundary conditions, etc.

  `COMPILES`: example compiles (for default parameters)

  `RUNS`: example runs (for default parameters)

  `CONVERGES`: no convergence issues (for default parameters, results not plausible)

  `PLAUSIBLE`: results look sensible (for default parameters)

  `VALIDATED`: for all parameter sets it gives the correct results as compared to CHeart/Abaqus/analytical solution (includes visualisation scripts, run scripts, comparison scripts, documentation!, . . .)

  Move progress Google-document into PDF-document

  Make from top directory - ensure all `run_example.sh` scripts are working as intended

4 IMMEDIATELY AFTER FRIDAY

- Move tags `CONVERGE, PLAUSIBLE` to `VALIDATED`
- Add GitHub issue for all tests/tags; `VALIDATED` means issue closed, else issue open
- Everybody runs everything!
- Meeting with Oliver
- Meeting with Auckland

5 LONG-TERM

- `SMALL/BIG/PARALLEL` targets
- Add more examples/those which were on the agenda but not started
- Jenkins
  - test `SMALL/BIG/PARALLEL` targets
  - integrate with GitHub (pull-requests triggers Jenkins, merge on success)

6 DIFFUSION EQUATION

6.1 Equation in general form

The governing equation is,

$$\partial_t u + \nabla \cdot [\boldsymbol{\sigma} \nabla u] = f, \tag{1}$$

with conductivity tensor $\boldsymbol{\sigma}$. The conductivity tensor is,

- defined in material coordinates (fibre direction),
- diagonal,
- defined per element.

6.2 Example-0001 [VALIDATED]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

6.2.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0, 2] \times [0, 1], \tag{2}$$

with boundary conditions

$$u = 0 \qquad x = y = 0, \tag{3}$$
$$u = 1 \qquad x = 2, y = 1. \tag{4}$$

No material parameters to specify.

6.2.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0, 2] \times [0, 1] \times [0, 1], \tag{5}$$

with boundary conditions

$$u = 0 \qquad x = y = z = 0, \tag{6}$$
$$u = 1 \qquad x = 2, y = z = 1. \tag{7}$$

No material parameters to specify.

6.2.3 *Computational model*

- Commandline arguments are:

  float: length along x-direction
  float: length along y-direction
  float: length along z-direction (set to zero for 2D)
  integer: number of elements in x-direction
  integer: number of elements in y-direction
  integer: number of elements in z-direction (set to zero for 2D)
  interger: interpolation order (1: linear; 2: quadratic)
  integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

  2.0 1.0 0.0 2 1 0 1 0
  2.0 1.0 0.0 4 2 0 1 0
  2.0 1.0 0.0 8 4 0 1 0
  2.0 1.0 0.0 2 1 0 2 0
  2.0 1.0 0.0 4 2 0 2 0
  2.0 1.0 0.0 8 4 0 2 0
  2.0 1.0 0.0 2 1 0 1 1
  2.0 1.0 0.0 4 2 0 1 1
  2.0 1.0 0.0 8 4 0 1 1
  2.0 1.0 0.0 2 1 0 2 1

2.0 1.0 0.0 4 2 0 2 1

2.0 1.0 0.0 8 4 0 2 1

2.0 1.0 1.0 2 1 1 1 0

2.0 1.0 1.0 4 2 2 1 0

2.0 1.0 1.0 8 4 4 1 0

2.0 1.0 1.0 2 1 1 2 0

2.0 1.0 1.0 4 2 2 2 0

2.0 1.0 1.0 8 4 4 2 0

2.0 1.0 1.0 2 1 1 1 1

2.0 1.0 1.0 4 2 2 1 1

2.0 1.0 1.0 8 4 4 1 1

2.0 1.0 1.0 2 1 1 2 1

2.0 1.0 1.0 4 2 2 2 1

2.0 1.0 1.0 8 4 4 2 1

6.2.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

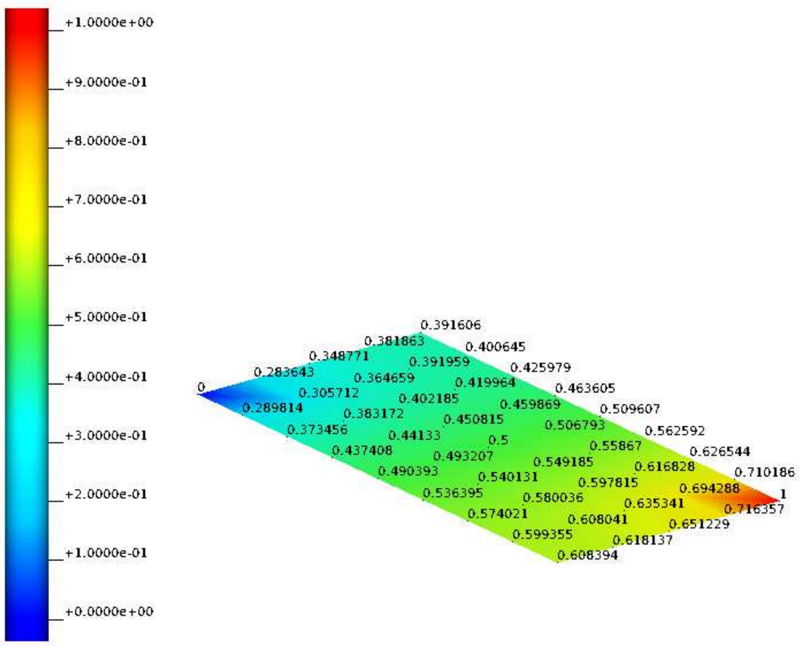

Figure 1: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

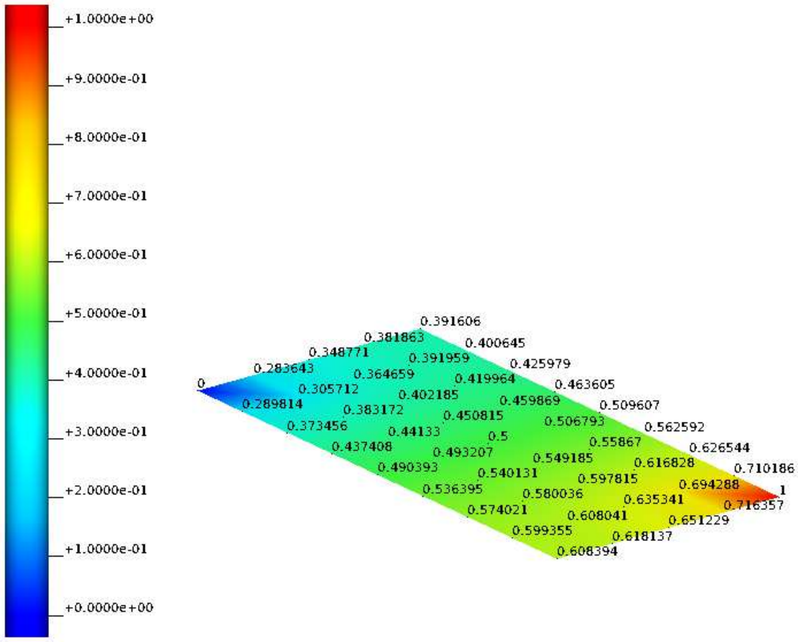

Figure 2: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

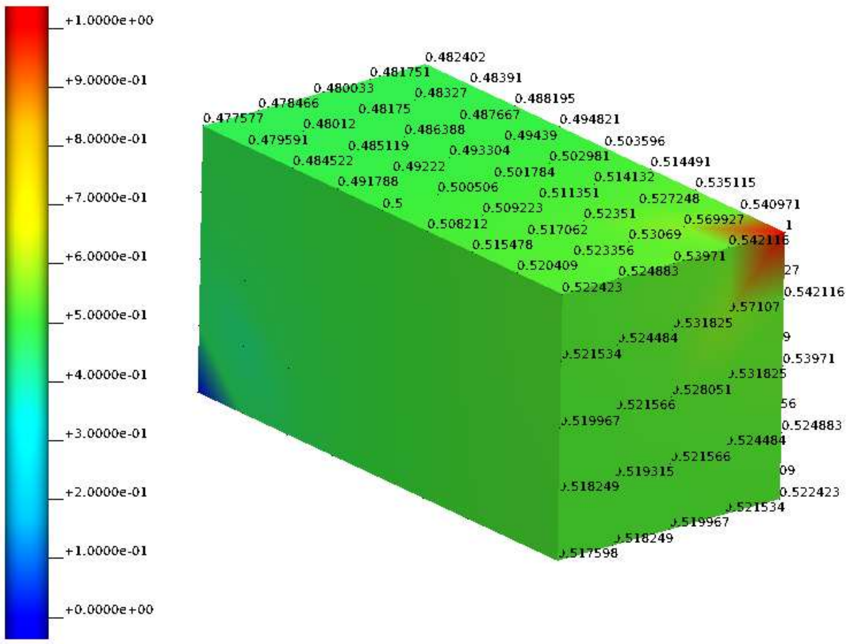

Figure 3: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

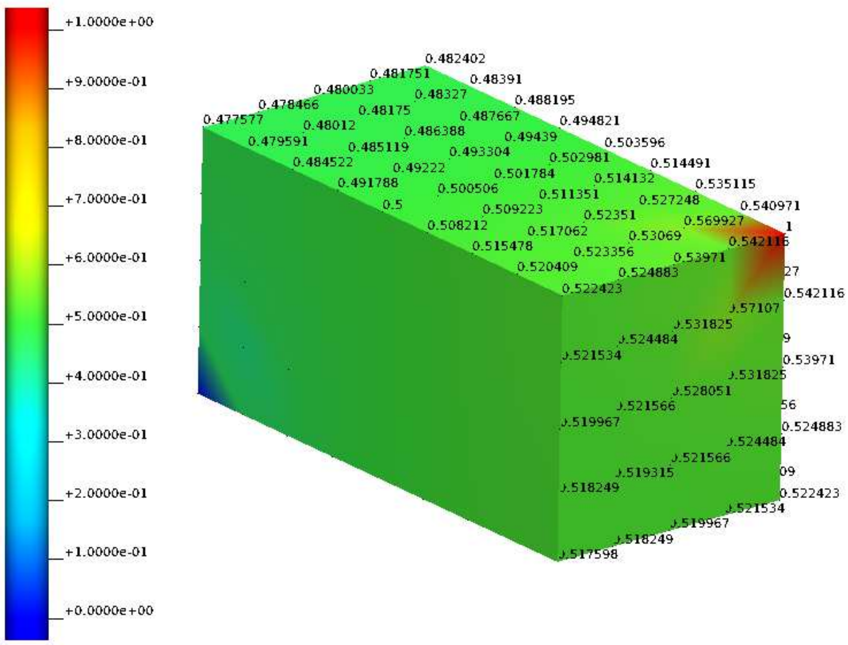

Figure 4: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

6.3 Example-0001-u [VALIDATED]

Example uses user-defined regular meshes in CHeart mesh format and solves a static problem, i.e., applies the boundary conditions in one step.

6.3.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0,2] \times [0,1], \tag{8}$$

with boundary conditions

$$u = 0 \qquad x = y = 0, \tag{9}$$
$$u = 1 \qquad x = 2, y = 1. \tag{10}$$

No material parameters to specify.

6.3.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0,2] \times [0,1] \times [0,1], \tag{11}$$

with boundary conditions

$$u = 0 \qquad x = y = z = 0, \tag{12}$$
$$u = 1 \qquad x = 2, y = z = 1. \tag{13}$$

No material parameters to specify.

6.3.3 *Computational model*

- Commandline arguments are:

  float: length along x-direction

  float: length along y-direction

  float: length along z-direction (set to zero for 2D)

  integer: number of elements in x-direction

  integer: number of elements in y-direction

  integer: number of elements in z-direction (set to zero for 2D)

  interger: interpolation order (1: linear; 2: quadratic)

  integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

  2.0 1.0 0.0 2 1 0 1 0

  2.0 1.0 0.0 4 2 0 1 0

  2.0 1.0 0.0 8 4 0 1 0

  2.0 1.0 0.0 2 1 0 2 0

  2.0 1.0 0.0 4 2 0 2 0

  2.0 1.0 0.0 8 4 0 2 0

  2.0 1.0 0.0 2 1 0 1 1

  2.0 1.0 0.0 4 2 0 1 1

  2.0 1.0 0.0 8 4 0 1 1

  2.0 1.0 0.0 2 1 0 2 1

2.0 1.0 0.0 4 2 0 2 1

2.0 1.0 0.0 8 4 0 2 1

2.0 1.0 1.0 2 1 1 1 0

2.0 1.0 1.0 4 2 2 1 0

2.0 1.0 1.0 8 4 4 1 0

2.0 1.0 1.0 2 1 1 2 0

2.0 1.0 1.0 4 2 2 2 0

2.0 1.0 1.0 8 4 4 2 0

2.0 1.0 1.0 2 1 1 1 1

2.0 1.0 1.0 4 2 2 1 1

2.0 1.0 1.0 8 4 4 1 1

2.0 1.0 1.0 2 1 1 2 1

2.0 1.0 1.0 4 2 2 2 1

2.0 1.0 1.0 8 4 4 2 1

- Note: Binary uses command line arguments to search for the relevant mesh files.

6.3.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

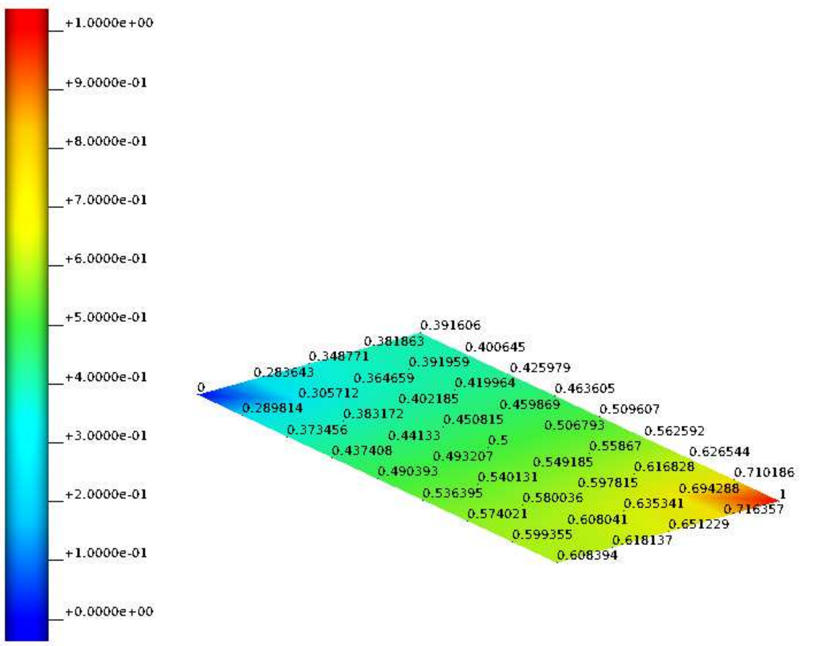

Figure 5: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

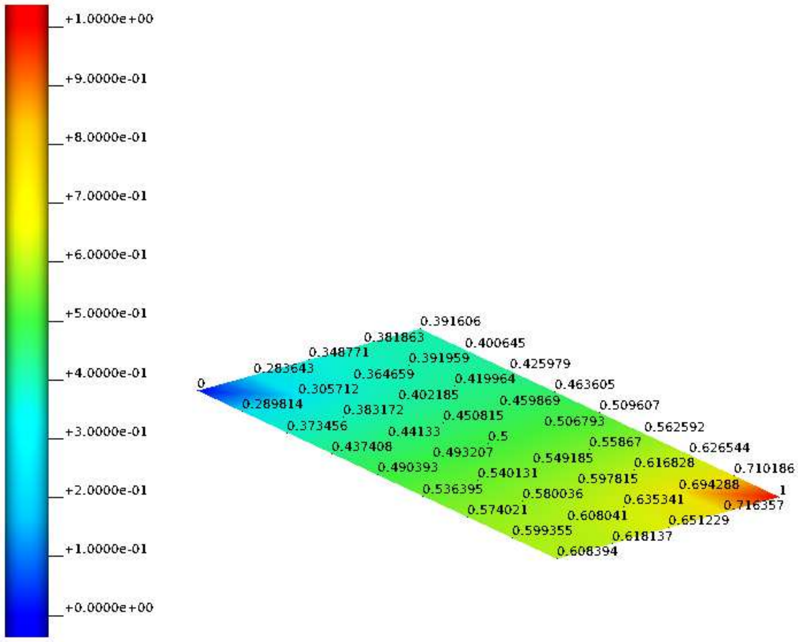

Figure 6: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

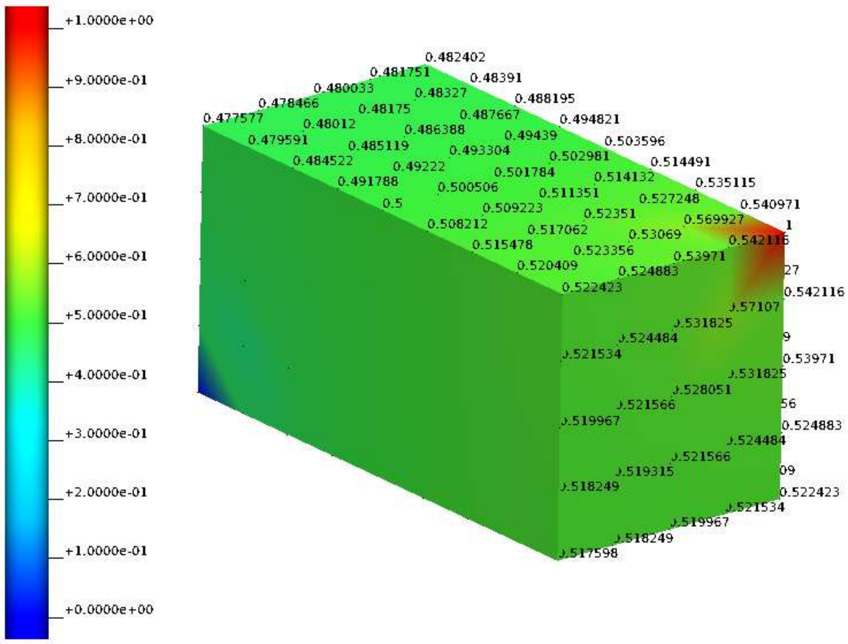

Figure 7: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

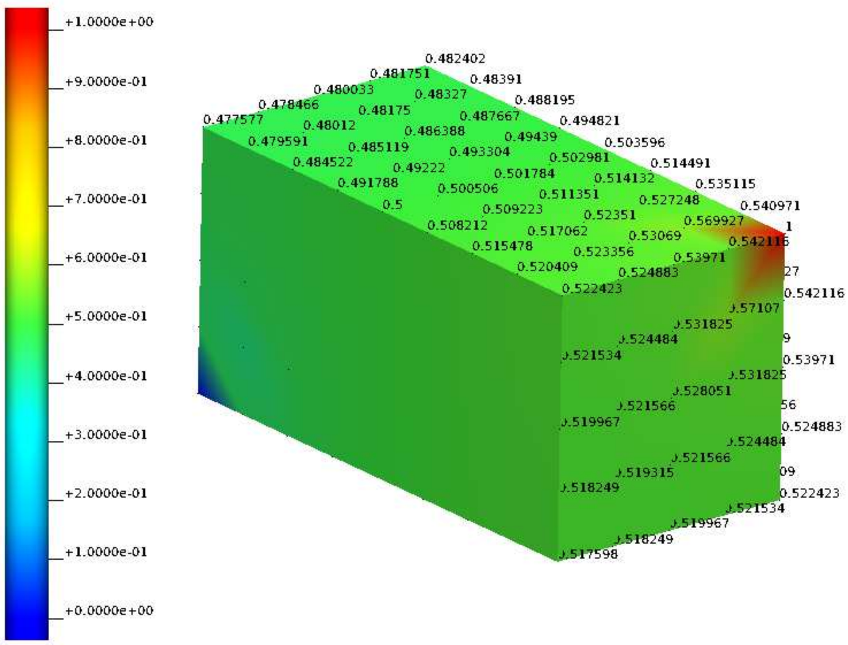

Figure 8: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

6.4 Example-0002 `[VALIDATED]`

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

6.4.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0,2] \times [0,1], \tag{14}$$

with boundary conditions

$$u = 15y \qquad x = 0, \tag{15}$$

$$u = 25 - 18y \qquad x = 2. \tag{16}$$

No material parameters to specify.

6.4.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0,2] \times [0,1] \times [0,1], \tag{17}$$

with boundary conditions

$$u = 15y \qquad x = 0, \tag{18}$$

$$u = 25 - 18y \qquad x = 2. \tag{19}$$

No material parameters to specify.

6.4.3 *Computational model*

- Commandline arguments are:

  float: length along x-direction
  float: length along y-direction
  float: length along z-direction (set to zero for 2D)
  integer: number of elements in x-direction
  integer: number of elements in y-direction
  integer: number of elements in z-direction (set to zero for 2D)
  interger: interpolation order (1: linear; 2: quadratic)
  integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

  2.0 1.0 0.0 2 1 0 1 0
  2.0 1.0 0.0 4 2 0 1 0
  2.0 1.0 0.0 8 4 0 1 0
  2.0 1.0 0.0 2 1 0 2 0
  2.0 1.0 0.0 4 2 0 2 0
  2.0 1.0 0.0 8 4 0 2 0
  2.0 1.0 0.0 2 1 0 1 1
  2.0 1.0 0.0 4 2 0 1 1
  2.0 1.0 0.0 8 4 0 1 1
  2.0 1.0 0.0 2 1 0 2 1

2.0 1.0 0.0 4 2 0 2 1

2.0 1.0 0.0 8 4 0 2 1

2.0 1.0 1.0 2 1 1 1 0

2.0 1.0 1.0 4 2 2 1 0

2.0 1.0 1.0 8 4 4 1 0

2.0 1.0 1.0 2 1 1 2 0

2.0 1.0 1.0 4 2 2 2 0

2.0 1.0 1.0 8 4 4 2 0

2.0 1.0 1.0 2 1 1 1 1

2.0 1.0 1.0 4 2 2 1 1

2.0 1.0 1.0 8 4 4 1 1

2.0 1.0 1.0 2 1 1 2 1

2.0 1.0 1.0 4 2 2 2 1

2.0 1.0 1.0 8 4 4 2 1

6.4.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

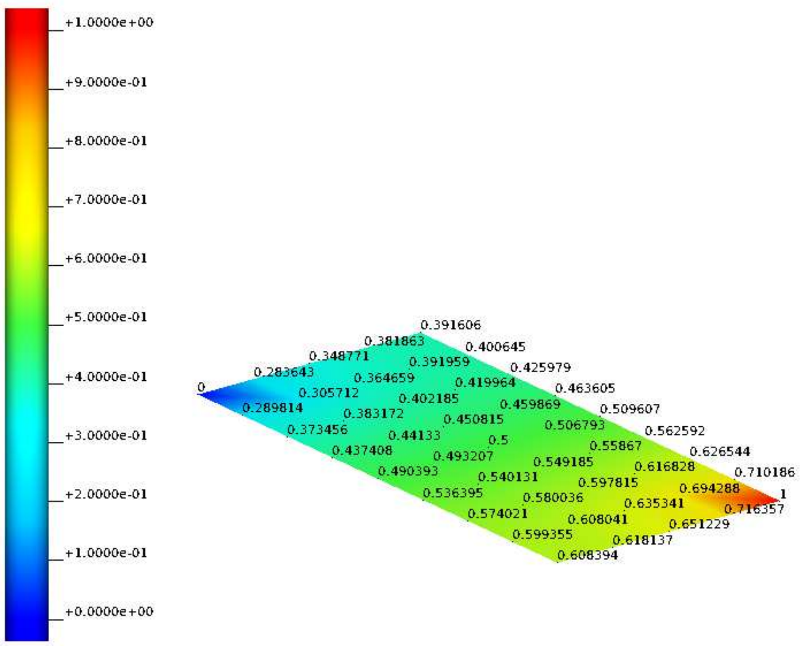

Figure 9: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

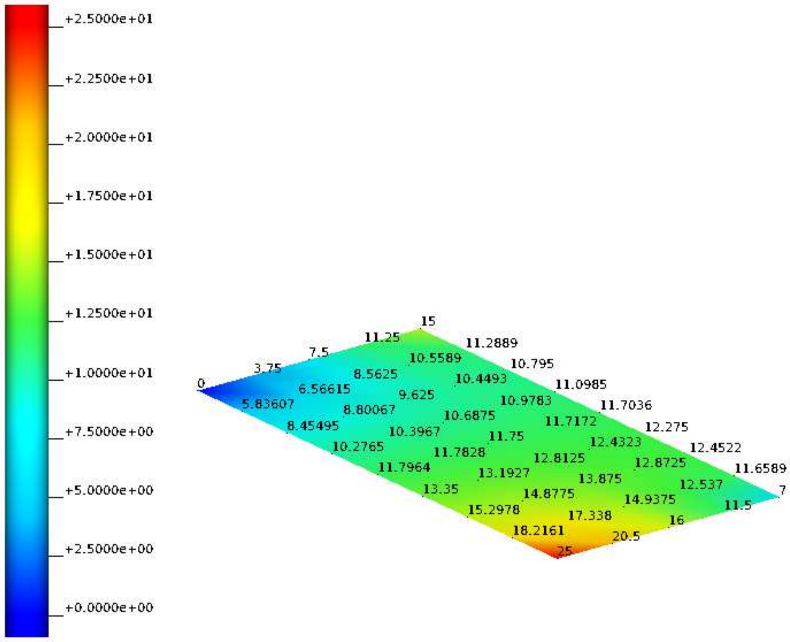

Figure 10: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

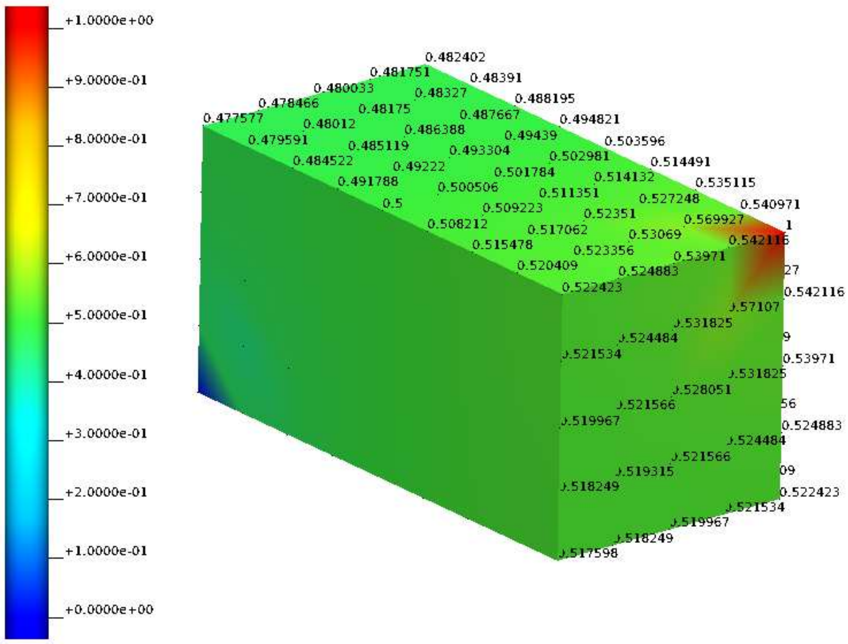

Figure 11: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

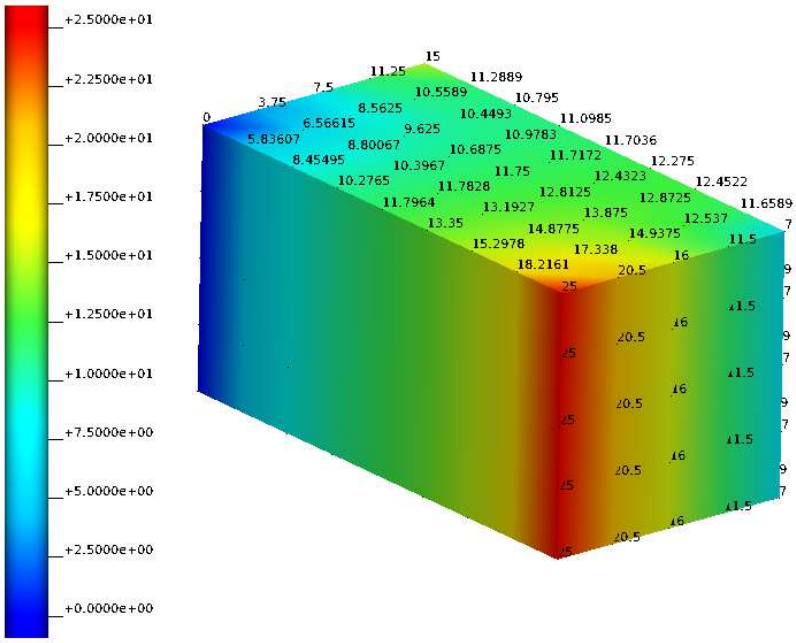

Figure 12: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

6.5 Example-0003

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

6.5.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0,2] \times [0,1], \tag{20}$$

with boundary conditions

$$u = 15y \qquad x = 0, \tag{21}$$
$$\partial_n u = 25 - 18y \qquad x = 2. \tag{22}$$

No material parameters to specify.

6.5.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \Omega = [0,2] \times [0,1] \times [0,1], \tag{23}$$

with boundary conditions

$$u = 15y \qquad x = 0, \tag{24}$$
$$\partial_n u = 25 - 18y \qquad x = 2. \tag{25}$$

No material parameters to specify.

6.5.3 *Computational model*

- Commandline arguments are:

  float: length along x-direction
  float: length along y-direction
  float: length along z-direction (set to zero for 2D)
  integer: number of elements in x-direction
  integer: number of elements in y-direction
  integer: number of elements in z-direction (set to zero for 2D)
  interger: interpolation order (1: linear; 2: quadratic)
  integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

  2.0 1.0 0.0 2 1 0 1 0
  2.0 1.0 0.0 4 2 0 1 0
  2.0 1.0 0.0 8 4 0 1 0
  2.0 1.0 0.0 2 1 0 2 0
  2.0 1.0 0.0 4 2 0 2 0
  2.0 1.0 0.0 8 4 0 2 0
  2.0 1.0 0.0 2 1 0 1 1
  2.0 1.0 0.0 4 2 0 1 1
  2.0 1.0 0.0 8 4 0 1 1
  2.0 1.0 0.0 2 1 0 2 1

2.0 1.0 0.0 4 2 0 2 1

2.0 1.0 0.0 8 4 0 2 1

2.0 1.0 1.0 2 1 1 1 0

2.0 1.0 1.0 4 2 2 1 0

2.0 1.0 1.0 8 4 4 1 0

2.0 1.0 1.0 2 1 1 2 0

2.0 1.0 1.0 4 2 2 2 0

2.0 1.0 1.0 8 4 4 2 0

2.0 1.0 1.0 2 1 1 1 1

2.0 1.0 1.0 4 2 2 1 1

2.0 1.0 1.0 8 4 4 1 1

2.0 1.0 1.0 2 1 1 2 1

2.0 1.0 1.0 4 2 2 2 1

2.0 1.0 1.0 8 4 4 2 1

6.5.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

Figure 13: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

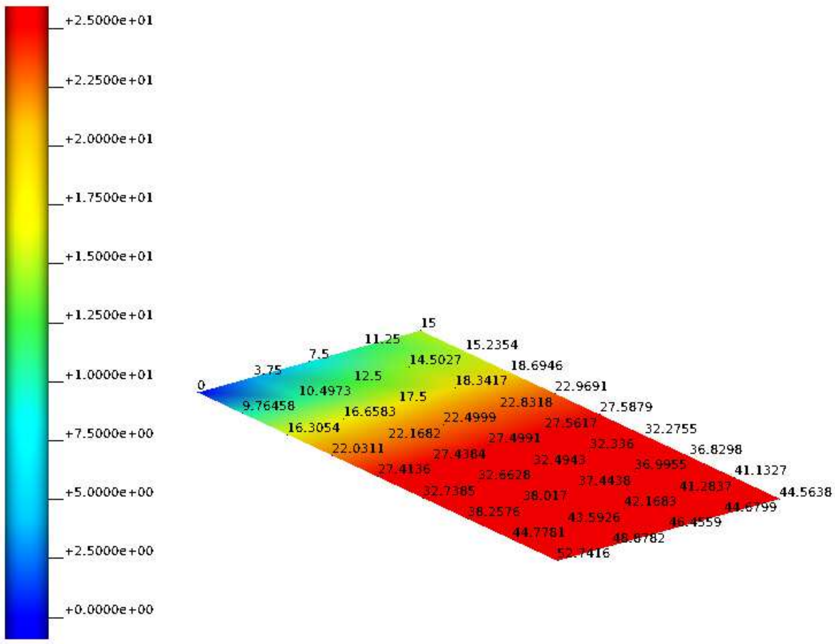

Figure 14: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

Figure 15: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

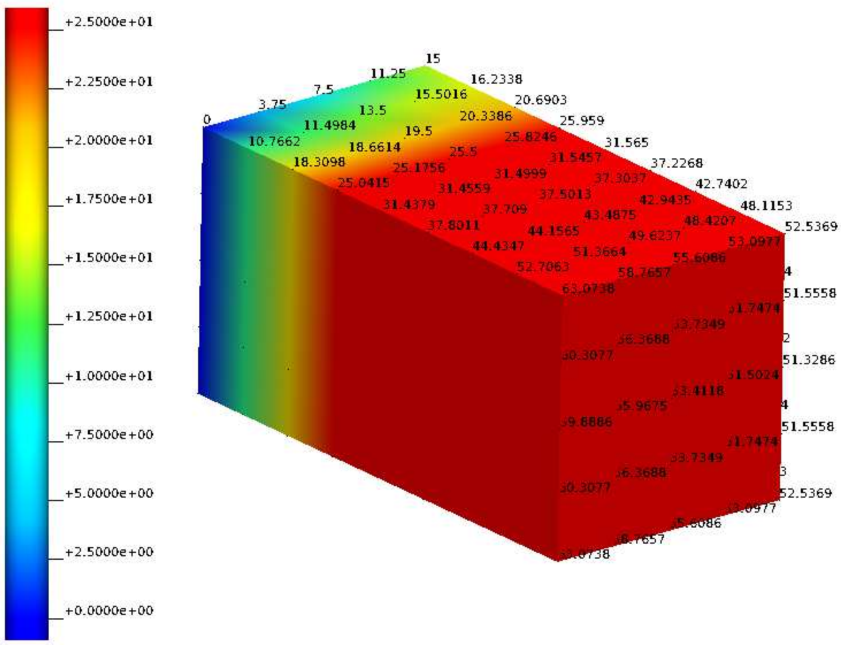

Figure 16: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

6.6 Example-0004 `[VALIDATED]`

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

6.6.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad\qquad \Omega = [0, 2] \times [0, 1], \tag{26}$$

with boundary conditions

$$u = 2.0e^{x} \cdot \cos(y) \qquad\qquad \text{on } \partial\Omega. \tag{27}$$

No material parameters to specify.

6.6.2 *Computational model*

- Commandline arguments are:

  integer: number of elements in x-direction

  integer: number of elements in y-direction

  integer: number of elements in z-direction (set to zero for 2D)

  interger: interpolation order (1: linear; 2: quadratic)

  integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

  4 2 0 1 0

  8 4 0 1 0

  2 1 0 2 0

  4 2 0 2 0

  8 4 0 2 0

  4 2 0 1 1

  8 4 0 1 1

  2 1 0 2 1

  4 2 0 2 1

  8 4 0 2 1

  100 50 0 1 0 (not tested yet..)

  100 50 0 2 0 (not tested yet..)

  100 50 0 1 1 (not tested yet..)

  100 50 0 2 1 (not tested yet..)

6.6.3 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

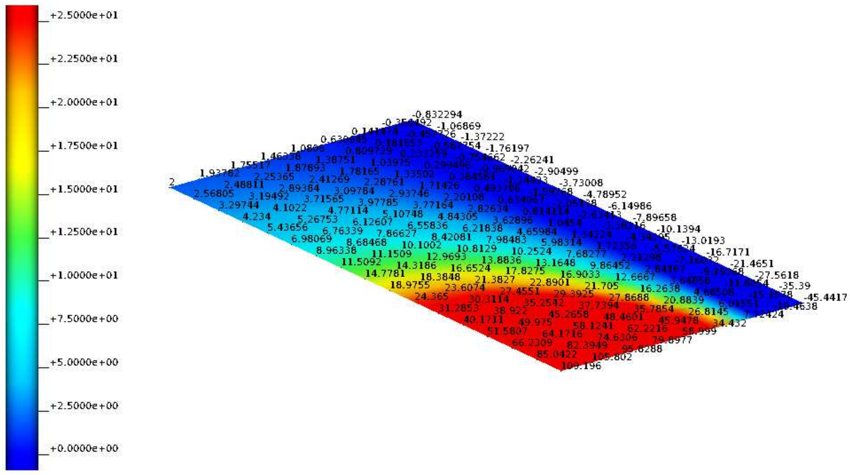

Figure 17: 2D results, iron reference w/ command line arguments [8 4 0 2 0].

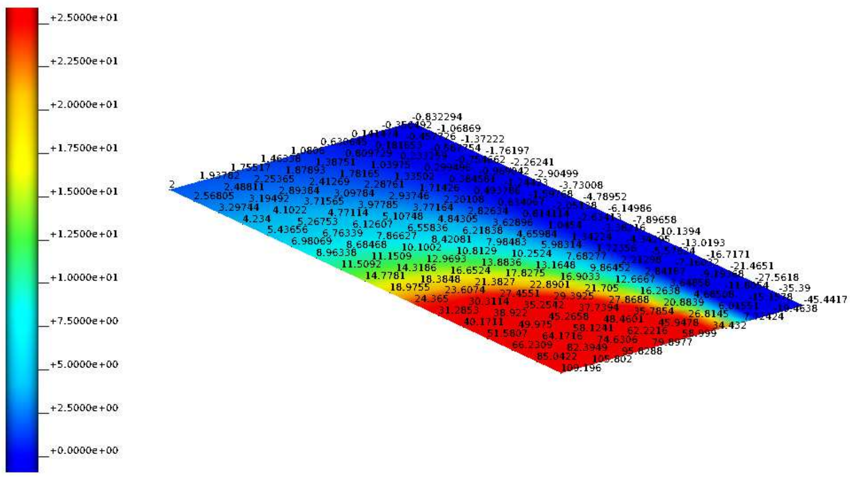

Figure 18: 2D results, current run w/ command line arguments [8 4 0 2 0].

6.7 Example-0011

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

6.7.1 *Mathematical model - 2D*

We solve the following scalar equation,

$$\nabla \cdot [\boldsymbol{\sigma} \nabla u] = 0 \qquad \Omega = [0,2] \times [0,1], \tag{28}$$

with boundary conditions

$$u = 0 \qquad x = y = 0, \tag{29}$$
$$u = 1 \qquad x = 2, y = 1. \tag{30}$$

The conductivity tensor is defined as,

$$\boldsymbol{\sigma}(x,t) = \boldsymbol{\sigma} = \mathbf{I}. \tag{31}$$

6.7.2 *Mathematical model - 3D*

We solve the following scalar equation,

$$\nabla \cdot [\boldsymbol{\sigma} \nabla u] = 0 \qquad \Omega = [0,2] \times [0,1] \times [0,1], \tag{32}$$

with boundary conditions

$$u = 0 \qquad x = y = z = 0, \tag{33}$$
$$u = 1 \qquad x = 2, y = z = 1. \tag{34}$$

The conductivity tensor is defined as,

$$\boldsymbol{\sigma}(x,t) = \boldsymbol{\sigma} = \mathbf{I}. \tag{35}$$

6.7.3 *Computational model*

- Commandline arguments are:

  float: length along x-direction

  float: length along y-direction

  float: length along z-direction (set to zero for 2D)

  integer: number of elements in x-direction

  integer: number of elements in y-direction

  integer: number of elements in z-direction (set to zero for 2D)

  integer: interpolation order (1: linear; 2: quadratic)

  integer: solver type (0: direct; 1: iterative)

  float: σ_{11}

  float: σ_{22}

  float: σ_{33} (ignored for 2D)

- Commandline arguments for tests are:

  2.0 1.0 0.0 2 1 0 1 0 1 1

  2.0 1.0 0.0 4 2 0 1 0 1 1

  2.0 1.0 0.0 8 4 0 1 0 1 1

  2.0 1.0 0.0 2 1 0 2 0 1 1

2.0 1.0 0.0 4 2 0 2 0 1 1

2.0 1.0 0.0 8 4 0 2 0 1 1

2.0 1.0 0.0 2 1 0 1 1 1 1

2.0 1.0 0.0 4 2 0 1 1 1 1

2.0 1.0 0.0 8 4 0 1 1 1 1

2.0 1.0 0.0 2 1 0 2 1 1 1

2.0 1.0 0.0 4 2 0 2 1 1 1

2.0 1.0 0.0 8 4 0 2 1 1 1

2.0 1.0 1.0 2 1 1 1 0 1 1 1

2.0 1.0 1.0 4 2 2 1 0 1 1 1

2.0 1.0 1.0 8 4 4 1 0 1 1 1

2.0 1.0 1.0 2 1 1 2 0 1 1 1

2.0 1.0 1.0 4 2 2 2 0 1 1 1

2.0 1.0 1.0 8 4 4 2 0 1 1 1

2.0 1.0 1.0 2 1 1 1 1 1 1 1

2.0 1.0 1.0 4 2 2 1 1 1 1 1

2.0 1.0 1.0 8 4 4 1 1 1 1 1

2.0 1.0 1.0 2 1 1 2 1 1 1 1

2.0 1.0 1.0 4 2 2 2 1 1 1 1

2.0 1.0 1.0 8 4 4 2 1 1 1 1

6.7.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

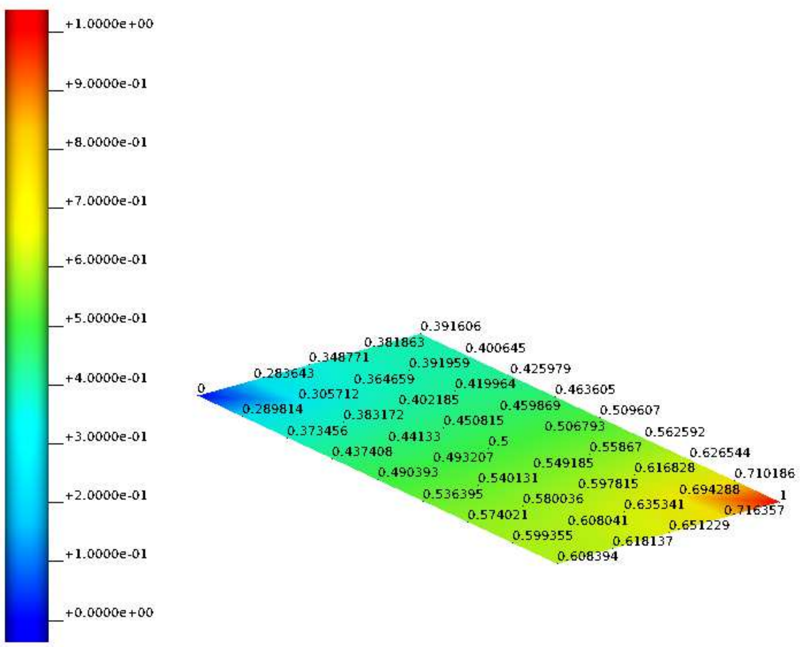

Figure 19: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 1 1].

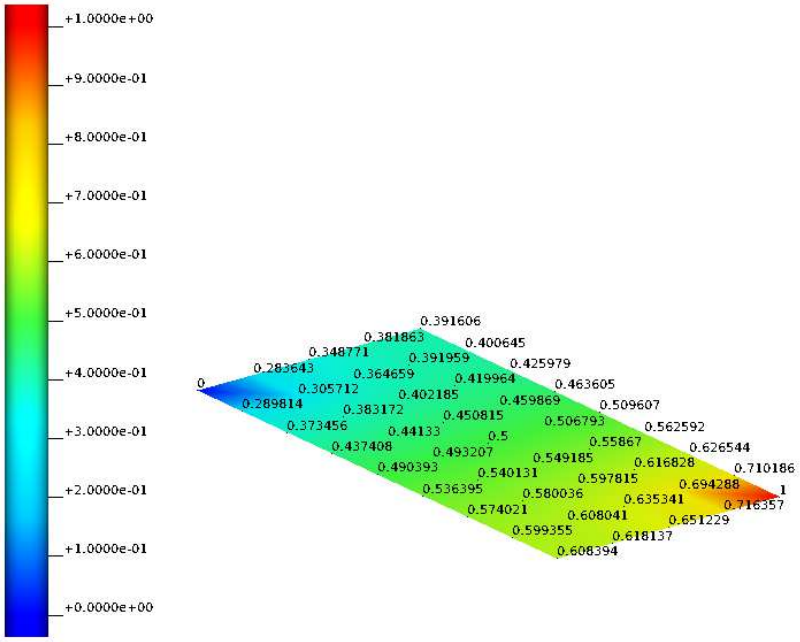

Figure 20: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 1 1].

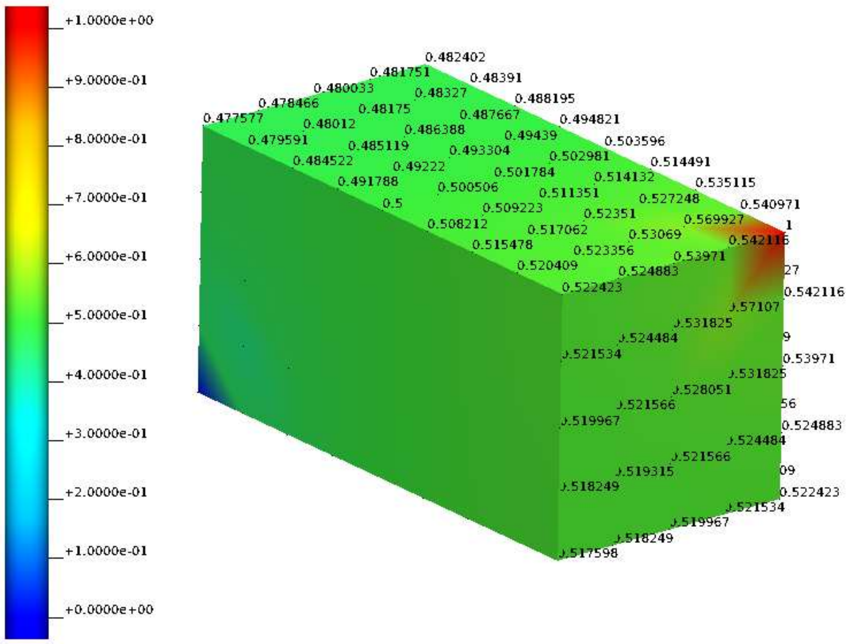

Figure 21: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0 1 1 1].

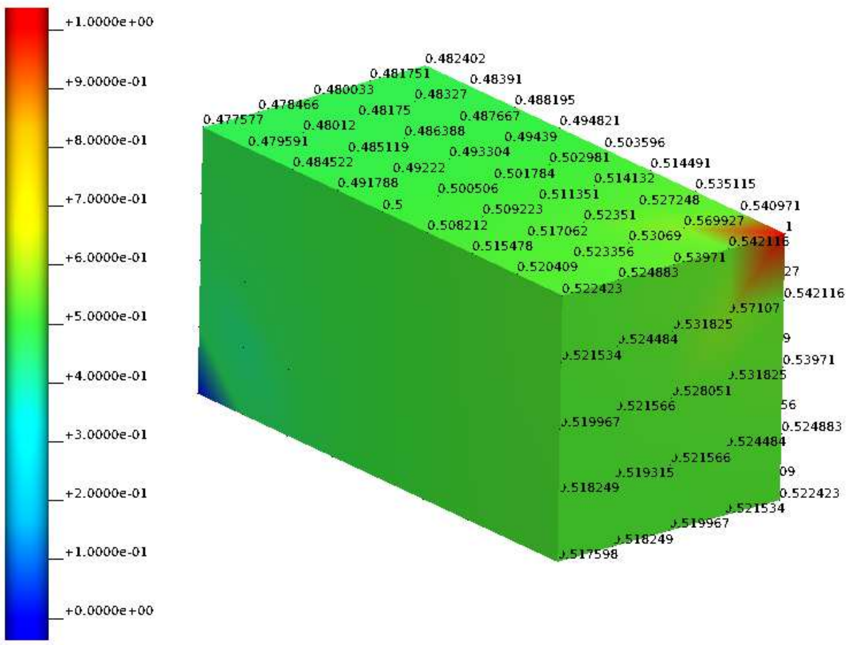

Figure 22: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0 1 1 1].

7 LINEAR ELASTICITY

7.1 Equation in general form

$$\partial_{tt}\mathbf{u} + \nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = \mathbf{f}(\mathbf{u}, t) \tag{36}$$

7.2 Example-0101 [PLAUSIBLE]

7.2.1 *Mathematical model*

We solve the following equation (both 2D and 3D domains are considered),

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = \mathbf{0} \qquad \Omega = [0, 160] \times [0, 120] \times [0, 120], t \in [0, 5], \tag{37}$$

with time step size $\Delta_t = 1$ and $\mathbf{u} = [u_x, u_y]$ in 2D $\mathbf{u} = [u_x, u_y, u_z]$ in 3D. The boundary conditions in 2D are given by

$$u_x = u_y = 0 \qquad x = y = 0, \tag{38}$$

$$u_x = 16 \qquad x = 160, \tag{39}$$

and in 3D by

$$u_x = u_y = u_z = 0 \qquad x = y = z = 0, \tag{40}$$

$$u_x = 16 \qquad x = 160. \tag{41}$$

The material parameters are

$$E = 10000\mathtt{MPa}, \tag{42}$$

$$\nu = 0.3, \tag{43}$$

$$\rho = 5 \times 10^{-9}\mathtt{tonne.mm}^3. \tag{44}$$

7.2.2 *Computational model*

- Commandline arguments are:
  - float: length along x-direction
  - float: length along y-direction
  - float: length along z-direction (set to zero for 2D)
  - integer: number of elements in x-direction
  - integer: number of elements in y-direction
  - integer: number of elements in z-direction (set to zero for 2D)
  - integer: interpolation order (1: linear; 2: quadratic)
  - integer: solver type (0: direct; 1: iterative)
  - float: elastic modulus
  - float: Poisson ratio
  - float: displacement percentage load
- Command line arguments for tests are:
  - 160 120 0 8 6 0 1 0 10000 0.3 0.05
  - 160 120 0 16 12 0 1 0 10000 0.3 0.05
  - 160 120 0 32 24 0 1 0 10000 0.3 0.05
  - 160 120 120 8 6 6 1 0 10000 0.3 0.05
  - 160 120 120 16 12 12 1 0 10000 0.3 0.05
  - 160 120 120 32 24 24 1 0 10000 0.3 0.05

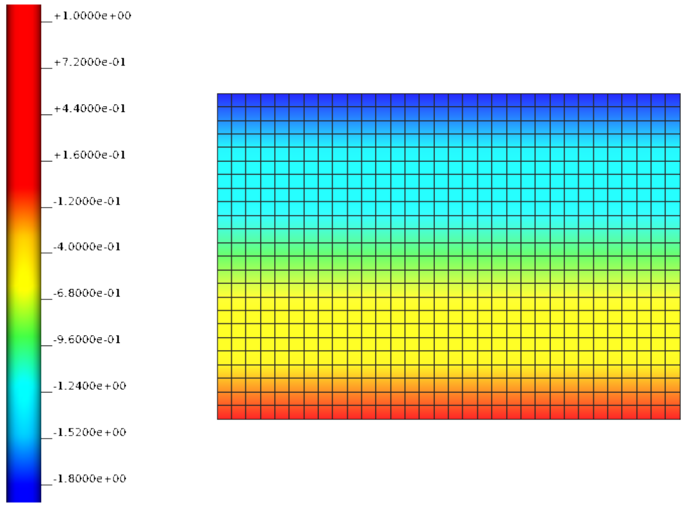

Figure 23: Results, iron 2D fine mesh.

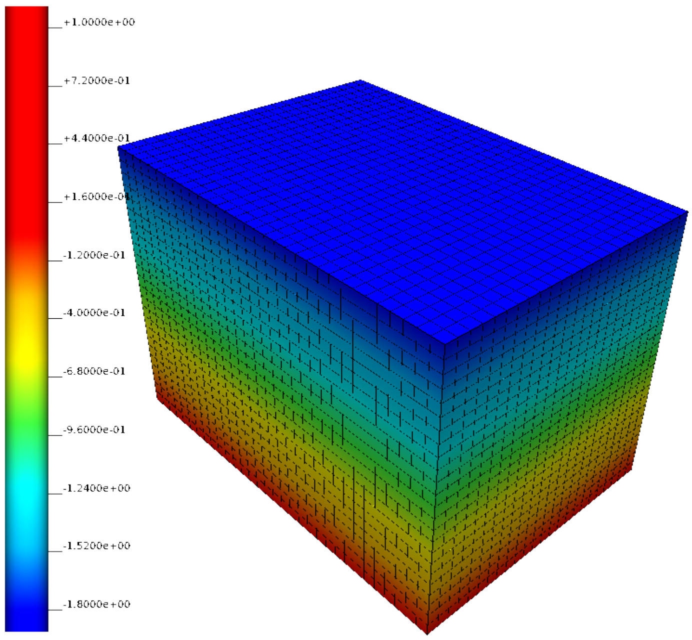

Figure 24: Results, iron 3D fine mesh.

7.2.3 *Results*

7.2.4 *Validation*

The iron results are compared to those from Abaqus (version 2017). The figures below show selected results from the validation simulations carried out in Abaqus and provide a qualitative validation. A quantitative validation was carried out by comparing the horizontal displacement u_y along the free-edge ($y = 120$ for 2D and $y = z = 120$ for 3D) and computing the L2-norm according to

$$\mathrm{L_2\text{-}norm} = \sum_{i=1}^{N} \sqrt{\left(u^i_{y,\mathrm{abaqus}} - u^i_{y,\mathrm{iron}}\right)^2}, \tag{45}$$

where N is the total number of nodes along the free-edge. The results over the mesh refinements are given in Table 2.

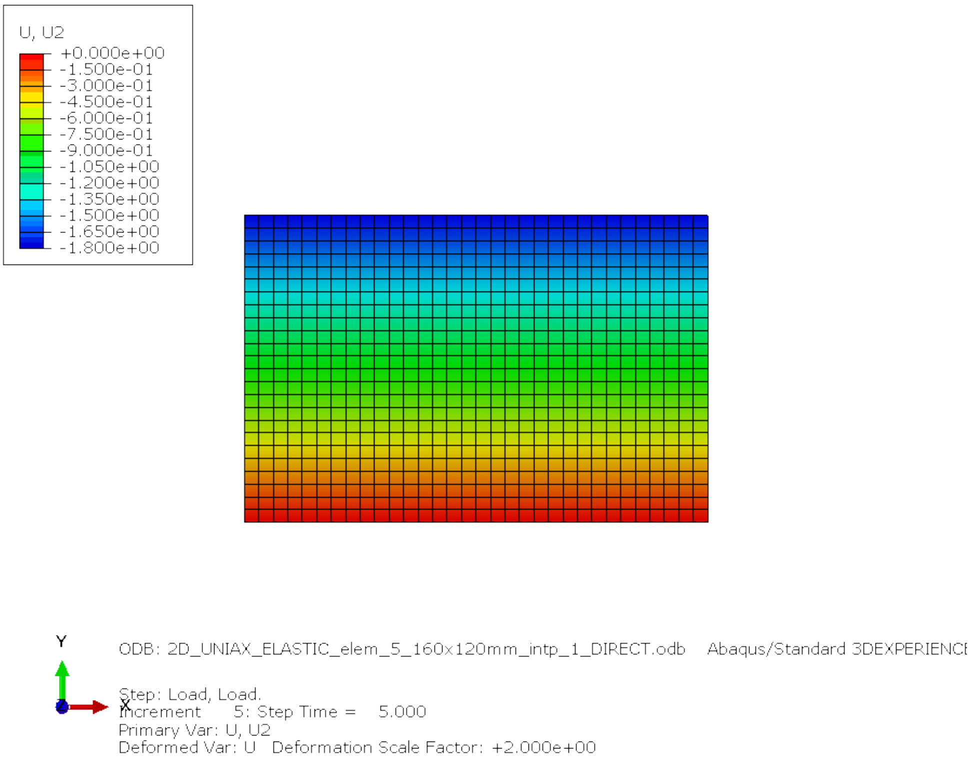

Figure 25: Results, Abaqus 2D fine mesh.

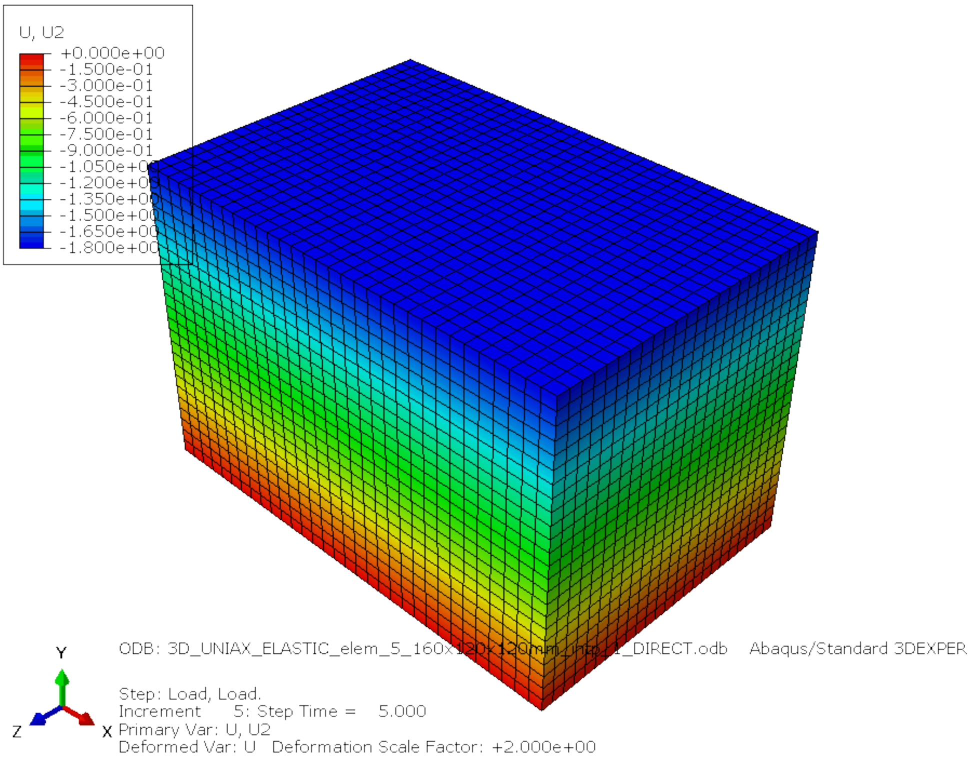

Figure 26: Results, abaqus 3D fine mesh.

Dimension	Mesh	L_2-`norm`	...
2D	Coarse	5.322×10^{-16}	...
2D	Medium	1.559×10^{-15}	...
2D	Fine	2.900×10^{-15}	...
3D	Coarse	3.071×10^{-17}	...
3D	Medium	2.125×10^{-17}	...
3D	Fine	2.924×10^{-17}	...

Table 2: Quantiative error between Abaqus 2017 and iron simulations for linear elastic uniaxial extenions

7.3 Example-0102 **[PLAUSIBLE]**

7.3.1 *Mathematical model*

We solve the following equation (both 2D and 3D domains are considered),

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = 0 \qquad \Omega = [0,160] \times [0,120] \times [0,120], t \in [0,5], \tag{46}$$

with time step size $\Delta_t = 1$ and $\mathbf{u} = [u_x, u_y]$ in 2D $\mathbf{u} = [u_x, u_y, u_z]$ in 3D. The boundary conditions in 2D are given by

$$u_x = u_y = 0 \qquad y = 0, \tag{47}$$
$$u_y = 8 \qquad x = 160, \tag{48}$$

and in 3D by

$$u_x = u_z = 0 \qquad x = 0, \tag{49}$$
$$u_y = 0 \qquad y = 0, \tag{50}$$
$$u_x = 160 \qquad x = 160, \tag{51}$$
$$u_y = 8 \qquad x = 160. \tag{52}$$

The material parameters are

$$E = 10000\texttt{MPa}, \tag{53}$$
$$\nu = 0.3, \tag{54}$$
$$\rho = 5 \times 10^{-9}\texttt{tonne.mm}^3. \tag{55}$$

7.3.2 *Computational model*

- Commandline arguments are:
  float: length along x-direction
  float: length along y-direction
  float: length along z-direction (set to zero for 2D)
  integer: number of elements in x-direction
  integer: number of elements in y-direction
  integer: number of elements in z-direction (set to zero for 2D)
  integer: interpolation order (1: linear; 2: quadratic)
  integer: solver type (0: direct; 1: iterative)
  float: elastic modulus
  float: Poisson ratio
  float: displacement percentage load
- Command line arguments for tests are:
  160 120 0 8 6 0 1 0 10000 0.3 0.05
  160 120 0 16 12 0 1 0 10000 0.3 0.05
  160 120 0 32 24 0 1 0 10000 0.3 0.05
  160 120 120 8 6 6 1 0 10000 0.3 0.05
  160 120 120 16 12 12 1 0 10000 0.3 0.05
  160 120 120 32 24 24 1 0 10000 0.3 0.05

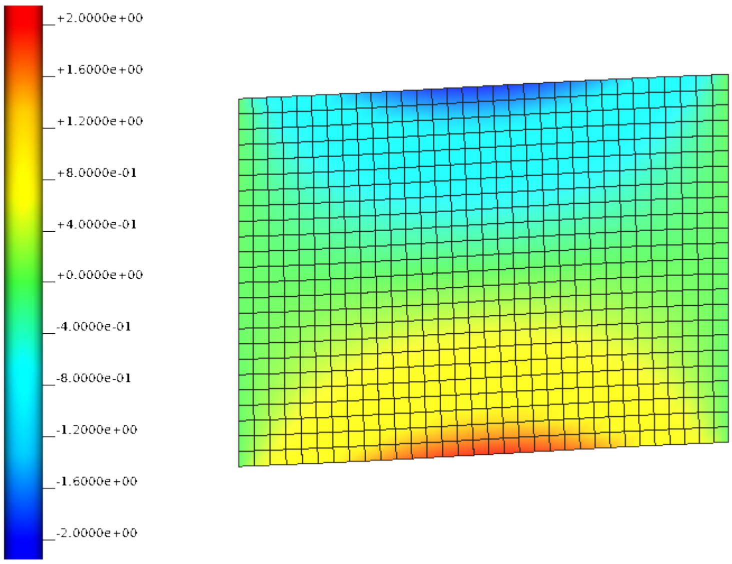

Figure 27: Results, iron 2D fine mesh.

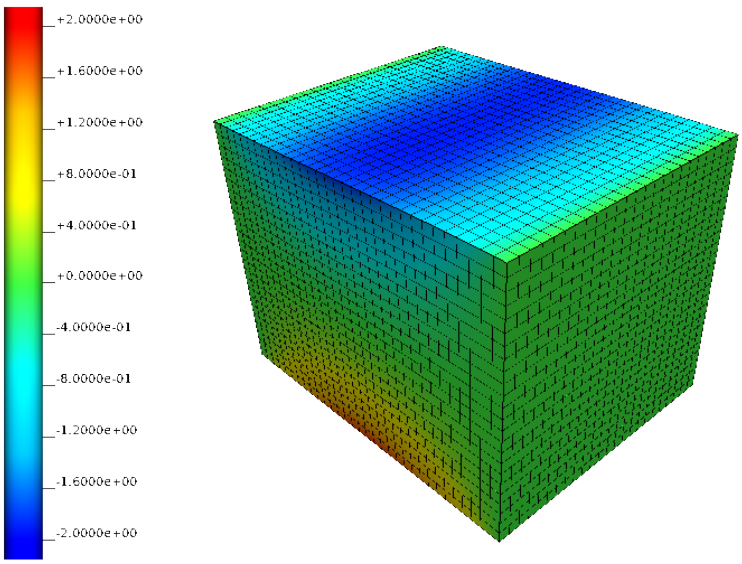

Figure 28: Results, iron 3D fine mesh.

7.3.3 *Results*

7.3.4 *Validation*

The iron results are compared to those from Abaqus (version 2017). The figures below show selected results from the validation simulations carried out in Abaqus and provide a qualitative validation. A quantitative validation was carried out by comparing the horizontal displacement u_x along the free-edge ($y = 120$ for 2D and $y = z = 120$ for 3D) and computing the L2-norm accodring to

$$L_2\texttt{-norm} = \sum_{i=1}^{N} \sqrt{\left(u^i_{y,\texttt{abaqus}} - u^i_{y,\texttt{iron}}\right)^2}, \tag{56}$$

where N is the total number of nodes along the free-edge. The results over the mesh refinements are given in Table 2.

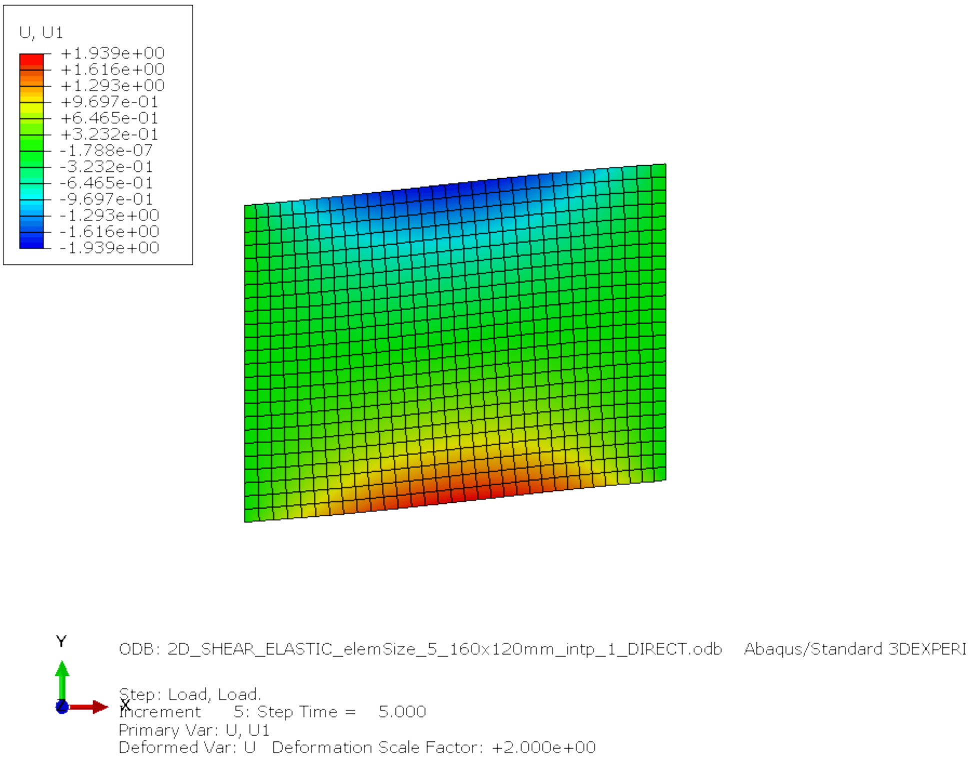

Figure 29: Results, Abaqus 2D fine mesh.

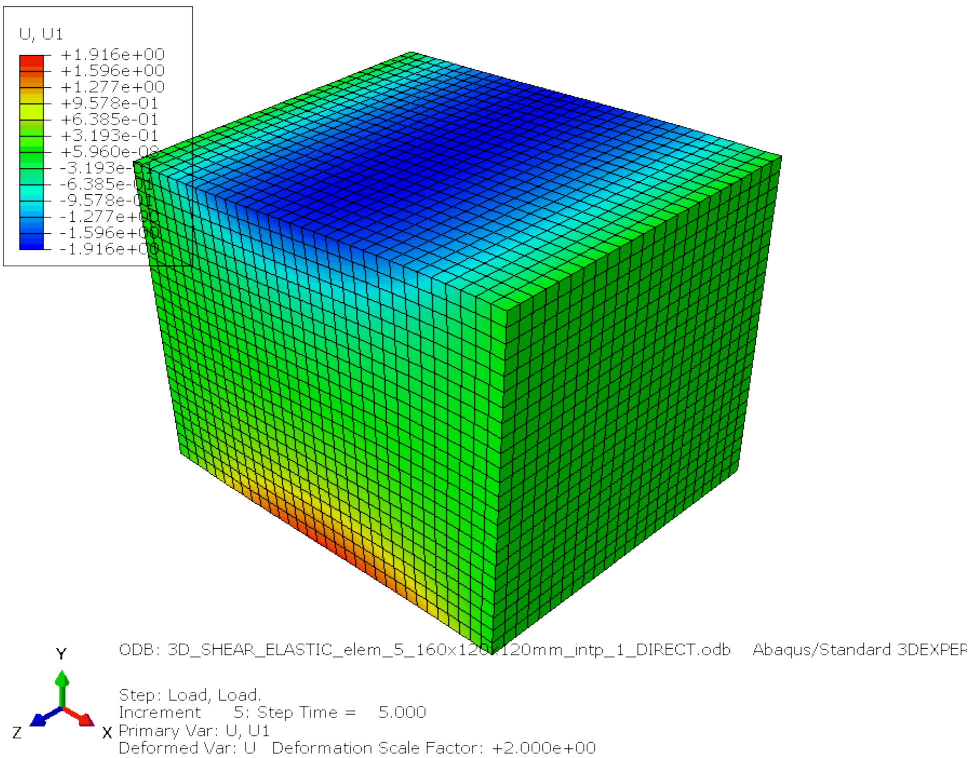

Figure 30: Results, abaqus 3D fine mesh.

Dimension	Mesh	L_2-`norm`	...
2D	Coarse	6.696×10^{-3}	...
2D	Medium	1.273×10^{-3}	...
2D	Fine	2.489×10^{-4}	...
3D	Coarse	4.234×10^{-4}	...
3D	Medium	4.184×10^{-5}	...
3D	Fine	3.781×10^{-6}	...

Table 3: Quantiative error between Abaqus 2017 and iron simulations for linear elastic shear

8 FINITE ELASTICITY

9 NAVIER–STOKES FLOW

9.1 Equation in general form

$$\partial_t(\rho\boldsymbol{v}) + \nabla\cdot(\rho\boldsymbol{v}\otimes\boldsymbol{v} + p\mathbf{I}) = \rho\mathbf{f} \tag{57}$$

9.2 Example-0302-u

Example uses user-defined simplex meshes in CHeart mesh format with quadratic/linear interpolation for velocity/pressure and solves a dynamic problem.

Setup is the well-known lid-driven cavity problem on the unit square or unit cube in two and three dimensions

9.2.1 *Mathematical model - 2D*

We solve the incompressible Navier-Stokes equation,

$$\partial_t(\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v} + p\mathbf{I}) = \rho \mathbf{f} \qquad \Omega = [0,1] \times [0,1], \tag{58}$$

$$\nabla \cdot \boldsymbol{v} = 0, \tag{59}$$

with boundary conditions

$$\boldsymbol{v} = 0 \qquad x = 0, \tag{60}$$

$$\boldsymbol{v} = 0 \qquad x = 1, \tag{61}$$

$$\boldsymbol{v} = 0 \qquad y = 0, \tag{62}$$

$$\boldsymbol{v} = [1,0]^T \qquad y = 1. \tag{63}$$

Density $\rho = 1$, viscosity $\mu = 0.0025$. Thus, Reynolds number $Re = 400$.

9.2.2 *Mathematical model - 3D*

We solve the incompressible Navier-Stokes equation,

$$\partial_t(\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v} + p\mathbf{I}) = \rho \mathbf{f} \qquad \Omega = [0,1] \times [0,1] \times [0,1], \tag{64}$$

$$\nabla \cdot \boldsymbol{v} = 0, \tag{65}$$

with boundary conditions

$$\boldsymbol{v} = 0 \qquad x = 0, \tag{66}$$

$$\boldsymbol{v} = 0 \qquad x = 1, \tag{67}$$

$$\boldsymbol{v} = 0 \qquad y = 0, \tag{68}$$

$$\boldsymbol{v} = [1,0]^T \qquad y = 1, \tag{69}$$

$$\boldsymbol{v} = 0 \qquad z = 0, \tag{70}$$

$$\boldsymbol{v} = 0 \qquad z = 1. \tag{71}$$

Density $\rho = 1$, viscosity $\mu = 0.01$. Thus, Reynolds number $Re = 100$.

9.2.3 *Computational model*

- Commandline arguments are:

  integer: number of dimensions (2: 2D, 3: 3D

  integer: mesh refinement level (1, 2, 3, ...)

  float: start time

  float: stop time

  float: time step size

  float: density

  float: viscosity

  integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

  2 1 0.0 0.01 0.001 1.0 0.0025 0

- Note: Binary uses command line arguments to search for the relevant mesh files.

9.2.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

10 MONODOMAIN

11 CELLML MODEL

REFERENCES

[1] Chris Bradley, Andy Bowery, Randall Britten, Vincent Budelmann, Oscar Camara, Richard Christie, Andrew Cookson, Alejandro F Frangi, Thiranja Babarenda Gamage, Thomas Heidlauf, et al. Opencmiss: a multi-physics & multi-scale computational infrastructure for the vph/physiome project. *Progress in biophysics and molecular biology*, 107(1):32–47, 2011.